Traducción del alfabeto dactilológico argentino al español utilizando redes neuronales convolucionales

Abstract. En Argentina, se estima que aproximadamente el 2% de la población padece algún tipo de incapacidad para oír (hipoacusia), este valor representa el 18% [1] del total de personas con discapacidades en el país.

Frente a esta problemática, se propone el desarrollo de un sistema que sea capaz de traducir en tiempo real el alfabeto dactilológico perteneciente a la Lengua de Señas Argentina a subtítulos en idioma español. De este modo se espera reducir la brecha comunicacional existente actualmente entre personas que padecen algún tipo de sordera en Argentina y aquellas que no comprenden su lengua de señas.

Se utilizaron técnicas de aprendizaje automático orientadas a la visión por computadora para desarrollar un modelo que reciba como entrada imágenes, e indique de qué señas pertenecientes al alfabeto dactilológico argentino se tratan. Para lograrlo, se construyó un "*dataset*" etiquetado que permitió el entrenamiento del modelo mencionado.

Empleando guantes, se logró un *Accuracy* de 0.9652 en el conjunto de datos de validación y de 0.9584 en el conjunto de test. Sin guantes se alcanzó un *Accuracy* de 0.8831 sobre el conjunto de validación y de 0.8205 en el conjunto de test. En ambos casos se utilizaron Redes Neuronales Convolucionales.

Keywords: Hipoacusia, Lengua de Señas Argentina, Inteligencia Artificial, Redes Neuronales Convolucionales.

1 Introducción.

"Traducción del alfabeto dactilológico argentino al español utilizando redes neuronales convolucionales" es un trabajo de investigación y desarrollo propuesto como Trabajo Final de Grado para la carrera "Ingeniería en Informática" de la Universidad Católica de Santiago del Estero, Departamento Académico Rafaela.

Como campo de acción se optó por el de la comunicación entre personas que padecen hipoacusia y aquellas que desconocen el alfabeto dactilológico argentino. Este primer público, al igual que cualquier otro, necesita comunicarse y de esta necesidad surge la iniciativa de contribuir a subsanar los inconvenientes que se generan al no comprender o interpretar incorrectamente su alfabeto.

En la Argentina aproximadamente 750.000 [1] personas padecen algún tipo de sordera o hipoacusia, esta cifra representa el 18% del total de personas en el país que padecen algún tipo de discapacidad [1]. La Lengua de Señas Argentina constituye su medio de

[1] El cálculo del porcentaje y la cantidad de población argentina que padece hipoacusia, se realizó a partir del porcentaje de personas con discapacidad según el INDEC [2], teniendo en cuenta además el porcentaje de personas que padecen sordera [1] y la cantidad total de personas que habitan en el país [3].

comunicación, pero, en la actualidad, es bajo el porcentaje de la población que tiene conocimiento acerca de esta lengua, inclusive en centros médicos como se menciona en [1]. Se tomará como referencia el alfabeto dactilológico argentino, para permitir la traducción de la Lengua de Señas Argentina al idioma español.

Desde una perspectiva social e inclusiva, se busca que los conocimientos adquiridos contribuyan a mejorar las relaciones entre ambos tipos de interlocutores, diseñando un modelo de aprendizaje automático capaz de clasificar los caracteres pertenecientes al alfabeto dactilológico argentino.

La solución propuesta utiliza modelos basados en redes neuronales convolucionales entrenados a partir de conjuntos de datos creados para tal fin. A partir de aquí los términos "Dataset" y conjuntos de datos serán tratados como sinónimos.

2 Metodología de trabajo.

Se decide utilizar CRISP-DM, como metodología de trabajo por ser "un método probado para orientar […] trabajos de minería de datos." [4] "El ciclo vital del modelo contiene seis fases con flechas que indican las dependencias más importantes y frecuentes entre fases."

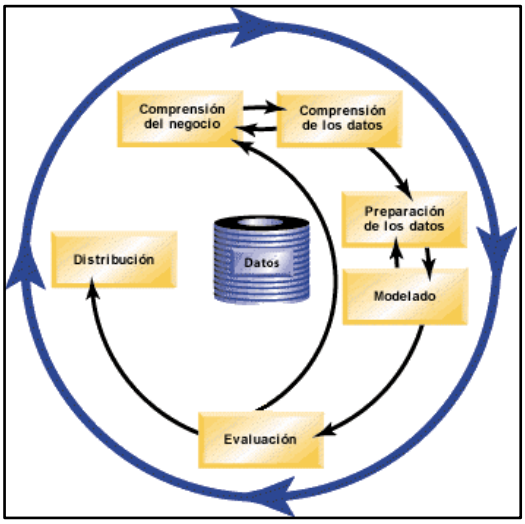

Fig. 1. Fases del ciclo de vida de la metodología CRISP-DM. Disponible en [4]

Fases de CRISP-DM:

1. Fase de comprensión del negocio o del problema.
2. Fase de comprensión de los datos.
3. Fase de preparación de los datos.
4. Fase de modelado.
5. Fase de evaluación.
6. Fase de implementación o distribución.

3 Desarrollo de la solución.

3.1 Fase de comprensión del problema.

Esta etapa fue abordada en el punto “Introducción” del presente trabajo donde se estudió la utilidad de la solución propuesta.

3.2 Fase de comprensión de los datos.

Recolección de los datos.

Para la recolección de los datos se construyó una herramienta llamada “Dataset Collector” que se encuentra descripta en el Apéndice 1. Ergo, se estableció un vínculo entre la Universidad Católica de Santiago del Estero (UCSE Departamento Académico Rafaela) y la Escuela Especial N° 2079 “Discapacitados Auditivos” de la ciudad de Rafaela solicitando colaboración al personal de la institución al momento de recolectar los datos. Estos/as voluntarios/as realizaron las señas pertenecientes al alfabeto dactilológico argentino, brindando datos genuinos y de confianza ya que algunas de las colaboradoras o supervisoras de los/as alumnos/as fueron docentes.

Fig. 2. Izquierda: Alumnos de la Escuela Especial Nro. 2.079 colaborando en la recolección de los datos. Derecha: docentes de la misma institución participando.

Datos recolectados.

Se contó con la participación de 19 personas de la institución para la recolección de datos. Se pidió a los/as voluntarios/as que realizaran 2 veces este proceso: una vez empleando guantes de látex color celeste y otra, sin. El motivo de la utilización de este accesorio se explica en la fase de “Pre-procesado de los datos”.

Aproximadamente el 58% de los videos fueron capturados a 30 FPS (fotogramas por segundo) con una duración de 6 segundos; el 42% restante, a 20 FPS con una duración de 5 segundos para obtener mayor variabilidad en el conjunto de datos.

Para transformar los videos en cada una de las imágenes que lo componen se construyó una herramienta (“Videos to images”) empleando Python, la cual se detalla en el Apéndice 2.

Table 1. Datos recolectados Institución Educativa.

	Cantidad total de personas	**Cantidad de personas: 6 segundos – 30 FPS**	**Cantidad de personas: 5 segundos – 20 FPS**	**Cantidad de letras**	**Imágenes tomadas**
Con guantes	19	11	8	29	80.620
Sin guantes	17	10	7	29	72.500
Imágenes totales					153.120

También se recolectaron 5.220 imágenes adicionales con guantes y 5.220 sin ellos, capturadas a partir de las señas realizadas por el autor de este trabajo aumentando el volumen disponible de datos.

Table 2. Datos recolectados totales.

Imágenes totales (con autor)	**163.560**

3.3 Fase de preparación de los datos.

Limpieza de los datos.

Al analizar los fotogramas obtenidos en la fase previa, fue posible notar que, en general, las primeras imágenes de cada letra no mostraron la seña que se pretendió realizar. Esto sucedió debido al tiempo que le insumió a cada intérprete cambiar de una letra a la siguiente. Frente a esta situación se realizó un proceso de limpieza a partir del que se identificaron cuáles fueron aquellas letras que no se hicieron correctamente por cada persona y así descartarlas junto a aquellos que las realizaron a destiempo o de forma incorrecta.

Por otro lado, se dejaron sin efecto las letras CH y la LL debido a que es posible conformar ambas empleando otras letras con las que ya se cuenta (C+H, L+L). Además, en el caso de la LL, puede que se generen ambigüedades con la letra L porque su representación es similar.

Cantidad de datos obtenidos luego de la limpieza:

Table 3. Cantidad de datos luego de la limpieza

	Cantidad personas	**Cantidad imágenes**
Sin guantes	18 personas	55.986
Con guantes	18 personas	56.399

Imágenes limpias: 112.385. Fueron descartadas 51.175 imágenes, cantidad que representa el 31,29% del total de los datos recolectados.

Pre-procesado de los datos.

Segmentación por color.

A partir del conjunto de datos de personas con guantes, y en base a lo planteado en [5], "Para llevar a cabo el proceso de filtrado se transformó el espacio de color RGB de la imagen a un espacio HSV (Hue, Saturation, Value)." En la Figura 3, se observa un ejemplo de las imágenes que componen el conjunto de datos de personas con guantes, en este caso representando la letra L.

Fig. 3. Representación de la letra L, empleando un guante de color celeste.

En la Figura 4, se visualiza a la izquierda una representación de la imagen de la Figura 3 en el espacio de color RGB, y a la derecha en el espacio de color HSV.

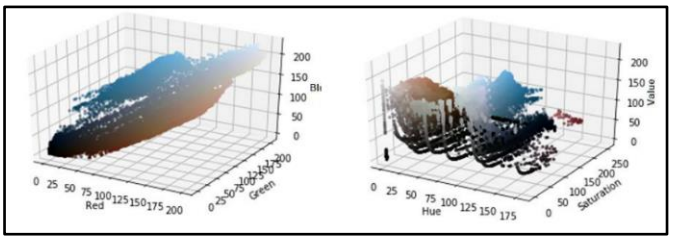

Fig. 4. Representación de la imagen de la Figura 3 en el espacio de color RGB a la izquierda y en el espacio de color HSV a la derecha.

No fue tarea sencilla segmentar la imagen de la Figura 3 en el espacio de color RGB ya que las tonalidades celestes se encuentran "dispersas" por todo el espacio, lo que complejizó separar los píxeles de este color de los restantes. Sin embargo, en el espacio de color HSV, se notó, como se menciona en [5] que “Este modelo permite identificar de un modo más robusto y sencillo el color a filtrar.”

Aplicando una máscara de color que intentó conservar las partes de la imagen que contenían los colores del guante que formaba la seña, fue posible segmentar la imagen de la Figura 3 como se puede observar en la parte izquierda de la Figura 5.

A partir de los contornos de la imagen segmentada, se obtuvieron las coordenadas pertenecientes a la misma que contienen la mayor área. Esta parte fue la que correspondía a la mano (recuadro verde de la Figura 5). Recortando la imagen segmentada a partir de las coordenadas previamente calculadas, se obtuvo una imagen de menor resolución que contenía únicamente la mano (segmentada del fondo) - Figura 5.

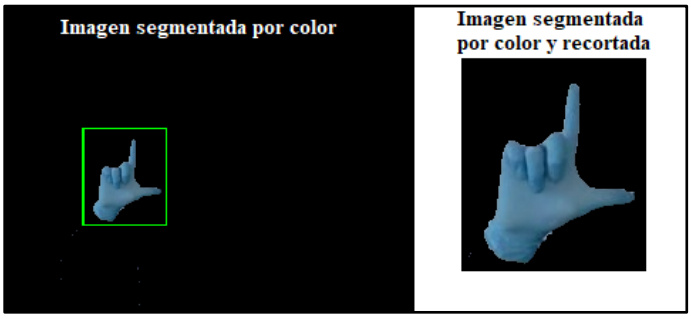

Fig. 5. Imagen de la Figura 3 segmentada por color a la izquierda. Imagen de la Figura 3 segmentada por color y recortada a la derecha.

Se desarrolló una herramienta (Apéndice 3) que realizó automáticamente el proceso descripto para todo el conjunto de imágenes (con guantes). Luego de aplicar dicha herramienta, se obtuvieron 56.399 imágenes con una resolución aproximada a 149x149 píxeles.

Recorte de bordes.

La resolución de las imágenes que conformaron el conjunto de datos sin guantes fue de 640x480 píxeles. Sin embargo, se determinó que aproximadamente el 44,44 % de la imagen correspondía a la persona realizando la seña, mientras que el 55,56 % restante pertenecía al fondo. Por ello, se recortaron las imágenes quedando sólo la parte que correspondía a la persona con el fin de mejorar el funcionamiento de los modelos.

Para llevar a cabo esta tarea se empleó el algoritmo de detección de bordes Canny, incluido en la librería OpenCV [6], a partir del cual se generó una nueva imagen representada por una matriz de números en la cual habría un “255” si el píxel corresponde a un borde, y un “0” en caso contrario. A partir de esta matriz fue posible obtener las coordenadas de los bordes de las personas en las imágenes. Luego se obtuvieron las coordenadas que encuadraban a la persona a partir del método boundingRect incluido en OpenCV [7] el cual recibió como entrada los bordes anteriores.

Posteriormente la imagen se cortó, dejando sólo la sección enmarcada por las coordenadas calculadas. Para que este proceso funcione correctamente, el fondo de la imagen debió ser una superficie de color sin la presencia de objetos que distraigan al sujeto principal (en las pruebas realizadas se utilizó una superficie blanca).

Para recortar automáticamente todas las imágenes del conjunto de datos sin guantes se creó una herramienta que, al aplicarla, las imágenes resultantes tuvieron una resolución promedio de 329x379 píxeles, representando aproximadamente el 40,59 % de su tamaño original.

En la Figura 6, se muestra a la izquierda un ejemplo de una imagen que compuso el conjunto de imágenes en tamaño original, mientras que a la derecha se ve la misma pero recortada por la herramienta desarrollada.

Fig. 6. Ejemplo de imagen sin recortar (izquierda) e imagen recortada por la herramienta desarrollada (derecha).

Segmentación por color de piel.

A partir de la idea propuesta en [8] se segmentó por color de piel a fin de obtener imágenes pre-procesadas que tenían únicamente la información relevante, pero sin la necesidad de utilizar los guantes previamente mencionados.

Para poder llevar a cabo el proceso, se realizó una segmentación por color en el espacio HSV de las imágenes sin guantes, aplicando dos máscaras de color que intentaron abarcar toda la escala cromática perteneciente a la piel humana.

Luego de dicha implementación, se obtuvieron imágenes que incluyeron ambos brazos y el rostro de la persona, omitiendo el resto de la imagen (ver Figura 7).

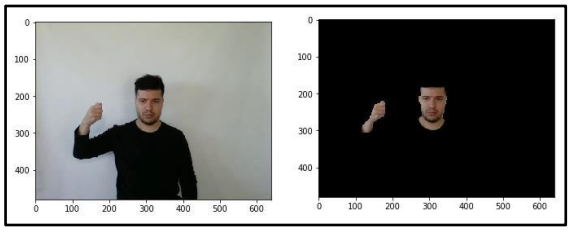

Fig. 7. Izquierda: imagen original. Derecha: imagen segmentada por color de piel.

Posteriormente se quiso detectar el rostro para ignorarlo en aquellas señas donde no se lo requería para representar la letra. Para lograrlo se utilizó la herramienta OpenCV y los detectores de rostros que incluye [9]. Luego de detectarlo, se lo eliminó, obteniendo como resultado una imagen como la que se observa en la Figura 8.

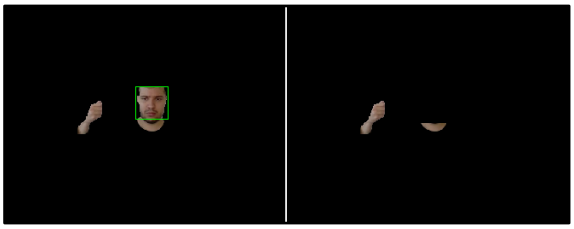

Fig. 8. Izquierda: rostro detectado en la imagen de la Figura 7. Derecha: Eliminación de rostro sobre la imagen de la Figura 7.

Luego, se decidió conservar sólo la porción de la imagen que contenía la seña. Para eso se obtuvieron los contornos de la imagen y se conservó la superficie que se encontraba delimitada dentro de estos contornos, superando un área preestablecida. A su vez, se le asignó prioridad a la esquina inferior izquierda de la imagen, puesto que las manos suelen ubicarse en dicho sector.

Finalmente, una vez detectada el área mencionada, se procedió a recortar la imagen. El resultado se observa en la parte izquierda de la Figura 9.

El proceso descripto se aplicó a todas las imágenes limpias sin guantes generando como resultado 2 datasets: uno que incluyó las imágenes segmentadas por color, como la que se observa en la parte izquierda de la Figura 9, y otro en el que se las recortó a partir de dónde se encontraba la mano, (aplicando el proceso antes mencionado). Ver Figura 9.

Fig. 9. Izquierda: seña de la imagen 7 segmentada por color. Derecha: seña de la imagen 7 recortada.

Armado de los “datasets”.

Prepare Dataset es una herramienta que se creó para armar los conjuntos de datos que fueron empleados por los modelos. Descripción en Apéndice 4.

Se utilizó la jerarquía de directorios generada por Prepare Dataset para usar Keras ImageDataGenerator [10], que facilitó la lectura de las imágenes y su correspondiente etiqueta (obtenida de cada imagen a partir del nombre de la carpeta que la contenía, por ejemplo, si la imagen se encontraba en la carpeta “B”, la etiqueta sería “B”). Además, permitió el aumento de datos.

Se aplicaron las siguientes técnicas de aumento de datos disponibles en Keras Preprocessing [10]:

- Rotación aleatoria. (rotation_range=5)
- Corte aleatorio de las imágenes en sentido antihorario. (shear_range=0.2)
- Zoom aleatorio. (zoom_range=0.2)

La cantidad de imágenes resultantes no variaron luego de aplicar las técnicas mencionadas, sino que aleatoriamente el modelo recibió imágenes con las modificaciones que dichas técnicas aplicaron sobre las mismas.

"Datasets" obtenidos.

Luego de la fase de limpieza y pre-procesado de los datos se contó con los siguientes "Datasets":

1. "Dataset" de imágenes sin guantes. Cuenta 55.933 imágenes.
2. "Dataset" de imágenes sin guantes recortadas. cuenta con 55.986 imágenes. Contiene sólo a la persona, recortando el fondo de las imágenes.
3. "Dataset" de imágenes segmentadas por color. Cuenta con 56.399 imágenes. Contiene las imágenes de las manos luego de realizar el proceso de segmentación por color a las imágenes dónde se utilizaron guantes.
4. "Dataset" de imágenes sin guantes recortadas 2. Cuenta con 55.365 imágenes. Contiene sólo la seña que se realizó en la imagen, esto incluye el brazo, la mano y el rostro en caso de ser necesario.
5. "Dataset" de imágenes sin guantes segmentadas por la piel. Cuenta con 55.211 imágenes. Es similar al "Dataset" 4, sólo que en este caso se cuenta con imágenes segmentadas por color, omitiendo la vestimenta y el fondo presente en las imágenes.

En la figura 10 se puede observar de izquierda a derecha una representación de la letra Q en cada "Dataset", ordenados de manera ascendente según su numeración.

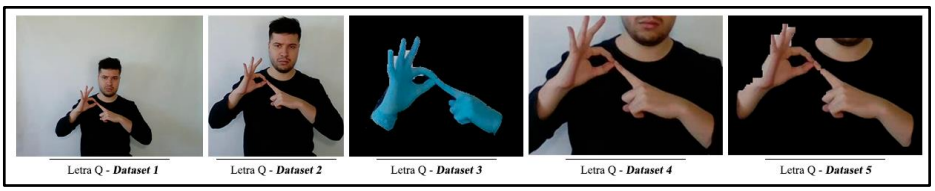

Fig. 10. Representación de la letra Q en cada "*Dataset"*, ordenados de manera ascendente de izquierda a derecha.

3.4 Fase de Modelado y Evaluación

Arquitecturas utilizadas.

Se utilizaron los siguientes modelos de Redes Neuronales Convolucionales: VGG-16 [11], ResNet-50 [12], Inception V3 [13] por tratarse de modelos de confianza debido a

que, como se observa en [14], fueron las arquitecturas que mejores resultados arrojaron en la métrica Accuracy. Además, han sido escogidas por su disponibilidad en el repositorio Keras Applications [15].

También, se utilizó en gran medida la arquitectura MobileNet, propuesta en [16] por su rapidez en las predicciones y su disponibilidad en Keras Applications [15].

Por último, se emplearon las siguientes arquitecturas de Redes Neuronales Convolucionales personalizadas:

- **Red Neuronal Convolucional: Personalizada 1.** Arquitectura propuesta: La red cuenta con 3 capas convolucionales seguidas de capas de agrupación *max pooling*, acopladas a una red completamente conectada compuesta por 2 capas densas de 1024 neuronas entre las cuales hay una capa de *Dropout*. La capa de salida cuenta con 27 neuronas. En total la red posee 9.560.155 parámetros, todos ellos entrenables.
- **Red Neuronal Convolucional: Personalizada 2**. Arquitectura propuesta: La red cuenta con 3 capas convolucionales seguidas de capas de agrupación *max pooling*, acopladas a una red completamente conectada compuesta por 2 capas densas: la primera con 512 neuronas y la segunda con 256 y entre ellas hay una capa de *Dropout*. La capa de salida cuenta con 27 neuronas. En total la red posee 4.426.331 parámetros, todos ellos entrenables.
- **Red Neuronal Convolucional: Personalizada 3.** Arquitectura propuesta: La red cuenta con 3 capas convolucionales seguidas de capas de agrupación *max pooling*, dónde entre cada capa de convolución - *max pooling* encontramos una capa de *Dropout*. Estas capas se encuentran acopladas a una red completamente conectada compuesta por 2 capas densas: la primera con 1024 neuronas y la segunda con 256, entre ellas hay una capa de *Dropout*. Además, entre las capas densas y las de *Dropout*, encontramos una capa de *BatchNormalization*, seguido de la capa de activación. En cada capa densa encontramos las siguientes regularizaciones: *kernel_regularizer* y *bias_regularizer* ambas del tipo l1. Estas regularizaciones se añadieron para intentar disminuir el sobre entrenamiento. La capa de salida cuenta con 27 neuronas. En total la red posee 8.757.339 parámetros, de los cuales 8.754.779 son entrenables.
- **Red Neuronal Convolucional: Personalizada 4.** Arquitectura propuesta: la arquitectura planteada es idéntica a perteneciente a la Red Neuronal Convolucional: Personalizada 2, solo que en este caso el tamaño de las entradas es de 149x149x3, dando como resultado 1.870.427 parámetros, todos ellos entrenables.

Métrica empleada.

Para evaluar el resultado de los modelos se utilizó la métrica Accuracy debido a que las clases que componen los conjuntos de datos se encuentran balanceadas, es decir existe aproximadamente la misma cantidad de imágenes de cada clase. Además, se escogió esta métrica por su interpretabilidad.

$$Accuracy = \frac{Número\ de\ predicciones\ correctas}{Número\ total\ de\ predicciones} \quad (1)$$

La definición formal de la métrica Accuracy de la ecuación 1 se encuentra disponible en [17].

En la Figura 11 se puede observar la cantidad promedio de imágenes por letra.

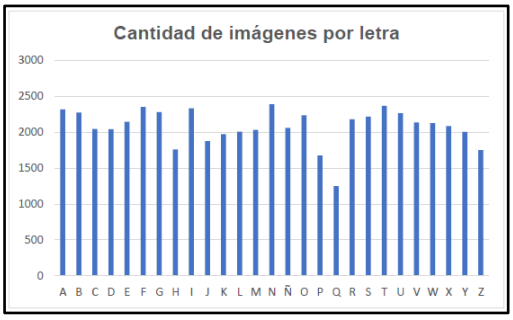

Fig. 11. Cantidad promedio de imágenes por letra.

Validación Cruzada.

Por cada uno de los conjuntos de datos con los que se trabajó, se utilizó una implementación propia del algoritmo de validación cruzada [18] para obtener resultados más robustos en aquellos modelos que arrojaron valores más altos de la métrica Accuracy.

La implementación de la validación cruzada fue realizada de la siguiente manera:

- A partir de cada uno de los conjuntos de datos mencionados en el apartado "'Datasets' obtenidos", se generaron 3 nuevos datasets, cada uno de los cuales se dividió en entrenamiento y validación. Se destinaron los datos pertenecientes a 15 personas para entrenamiento y los de 2 para validación. Lo que difiere entre cada uno de estos 3 nuevos datasets son las personas que se utilizaron para constituir cada conjunto de entrenamiento y cada conjunto de validación.
- Luego se entrenó cada modelo con los nuevos conjuntos de datos generados. Es decir, se utilizaron estos 3 nuevos datasets para entrenar cada modelo, en lugar de utilizar sólo el conjunto original.
- Como resultado final de la métrica para el conjunto de entrenamiento y para el de validación (por cada modelo/conjunto de datos de los mencionados en "'Datasets' obtenidos"), se utilizó la media aritmética de los resultados que arrojó el modelo empleando cada uno de estos 3 nuevos datasets para el conjunto de entrenamiento y validación respectivamente.
- Luego de realizar el entrenamiento aplicando validación cruzada, los modelos fueron entrenados nuevamente, pero esta vez utilizando la totalidad de los datos disponibles en los 3 *datasets*, es decir los de las 17 personas. Posteriormente se testearon empleando las imágenes pertenecientes a 1 persona, cuyos datos no se emplearon en el proceso de entrenamiento. Esto se realizó con el objetivo de evaluar el desempeño de cada modelo.

Experimentos realizados.

Para cada conjunto de datos se realizaron experimentos con diversas arquitecturas de Redes Neuronales Convolucionales:

Dataset imágenes sin guantes.

Para este conjunto de datos se experimentó con los modelos:

1. Red Neuronal Convolucional: Personalizada 1.
2. Red Neuronal Convolucional: Personalizada 2.
3. *MobileNet* – Validación Cruzada.

Dataset imágenes sin guantes recortadas.

Para este conjunto de datos se experimentó con los modelos:

1. Red Neuronal Convolucional: Personalizada 3.
2. *VGG-16: Transfer Learning*
3. *ResNet-50*
4. *MobileNet*
5. *Inception V3*
6. *MobileNet – Validación Cruzada*

Dataset imágenes segmentadas por color.

Para este conjunto de datos se experimentó con el modelo:

1. Red Neuronal Convolucional: Personalizada 4.
2. Red Neuronal Convolucional: Personalizada 4 – Validación Cruzada.

Dataset imágenes sin guantes recortadas 2.

Para este conjunto de datos se experimentó con el modelo:

1. *MobileNet* – Validación Cruzada.

Dataset imágenes sin guantes segmentadas por la piel.

Para este conjunto de datos se experimentó con el modelo:

1. *MobileNet* – Validación Cruzada.

Resultados obtenidos.

A continuación, se describen los mejores resultados obtenidos a partir de los experimentos sobre los conjuntos de datos mencionados en el apartado “’Datasets obtenidos”.
Resultados:

Table 4. Mejores resultados obtenidos a partir de los experimentos realizados.

Conjunto de datos	Cantidad Total de datos	Modelo	*Accuracy* Entrena-miento	*Accuracy* Validación	*Accuracy* Test
Imágenes sin guantes	55.933	MobileNet - CrossValidation	0,9959	0,8199	0,6485
Imágenes sin guantes recortadas	55.986	MobileNet - CrossValidation	0,9981	0,8831	0,8205
Imágenes segmen-tadas por color	56.399	Red Neuronal Convolucional – Personalizada 4 - CrossValidartion	0,9947	0,9652	0,9584
Imágenes sin guantes recortadas 2	55.365	MobileNet - CrossValidation	0,9977	0,9147	0,7547
Imágenes segmen-tadas por la piel	55.211	MobileNet - CrossValidation	0,9962	0,8898	0,8096

Conforme a esta información se concluyó que el conjunto de datos que logró los mejores resultados en la métrica Accuracy fue el de imágenes segmentadas por color. Este conjunto de datos provino luego de aplicar el proceso de pre-procesado: segmentación por color, a las imágenes que se tomaron utilizando guantes.

En segundo lugar, se ubicó el conjunto de datos de imágenes sin guantes recortadas, lo que indica que no fue necesario aplicar más pre-procesado que recortar las imágenes por sus bordes. También evidenció una mejora significativa frente a utilizar las imágenes tal cual fueron tomadas.

Para poder acceder al detalle completo de los resultados correspondientes a cada uno de los experimentos mencionados en el apartado “Experimentos realizados”, consultar el Apéndice 5.

3.5 Fase de implementación o distribución.

Se desarrolló una aplicación capaz de recibir un video y predecir la letra presente en cada uno de sus frames. El video que recibe puede ser un archivo o bien, uno capturado en tiempo real por la cámara web del ordenador donde se ejecute la aplicación.

En caso de que no se utilicen guantes, será condición necesaria utilizar ropa negra u oscura para que la aplicación el funcione correctamente.

En la Figura 12, a la derecha, se visualiza la aplicación identificando la letra L en un video en el que no se utilizaron guantes. Como se puede observar, el contorno de la persona se recuadró en color rojo (se empleó el mismo color para la palabra que se deletreó). En color amarillo se visualiza la letra que se hizo en un momento dado, junto a la probabilidad asociada a la confianza que el modelo arrojó sobre la predicción.

Una limitación de no utilizar guantes es que la aplicación sólo puede traducir una palabra.

Al emplear guantes, usar una remera negra u oscura no implica una condición necesaria, sino que el sujeto puede llevar otro tipo de prenda como se puede observar en la sección izquierda de la Figura 12. Además, la principal ventaja de la utilización de guantes es que la aplicación es capaz de distinguir cuándo finaliza una palabra y comienza otra a partir de la ausencia de la/s mano/s, permitiendo la traducción de más de una palabra.

Fig. 12. Izquierda: predicción utilizando guantes. Derecha: predicción sin utilizar guantes.

4 Conclusiones.

Los objetivos inherentes al proyecto han sido alcanzados, puesto que fue posible encontrar un modelo que traduzca el alfabeto dactilológico argentino en tiempo real. Para lograrlo se planteó una serie de experimentos que permitieron encontrar tal modelo. A lo largo del trabajo se mencionaron las pruebas realizadas y sus resultados, los cuales, en las condiciones óptimas, resultan favorables.

Sin embargo, pese a que los resultados obtenidos son alentadores, el presente trabajo no pretende dar lugar a una herramienta lista para su implementación en escenarios reales ya que posee limitaciones. Más bien, se busca dar un pantallazo sobre las posibilidades que las herramientas de aprendizaje automático y visión por ordenador bridan en un campo poco explorado en nuestro país. Tratando de incentivar a la investigación en este campo y de realizar un aporte que, a título personal, resulta indispensable. Para favorecer líneas de investigaciones futuras, se creó una extensa base de datos que puede resultar útil o servir de guía en posteriores trabajos.

Referencias

1. A. M. Cutria, F. A. Torresa, C. C. Riquelme, N. B. Cabrera, R. R. Sandoval, S. M. Vesconi, S. Di Lalla y L. Cordi, «Prevalencia de personas sordas que cuentan con un intérprete profesional de Lengua de Señas Argentina en la consulta médica de sus hijos,» p. 8, 2018.
2. INDEC, Estudio Nacional sobre el Perfil de las Personas con Discapacidad, 2018.

3. INDEC, «indec.gob.ar,» 2019. [En línea]. Available: https://www.indec.gob.ar/indec/web/Nivel3-Tema-2-41. [Último acceso: 5 Septiembre 2019].
4. IBM, «Manual CRISP-DM de IBM SPSS Modeler,» 1994.
5. F. Ronchetti, Reconocimiento de gestos dinámicos y su aplicación al lenguaje de señas, 2016.
6. o.-c. p. tutroals, «open-cv python tutroals,» [En línea]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html. [Último acceso: 9 Diciembre 2019].
7. opencv, «docs.opencv.org,» [En línea]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=boundingrect. [Último acceso: 9 Diciembre 2019].
8. P. Soffritti, «Handy, hand detection with OpenCV,» 2018 . [En línea]. Available: https://medium.com/@soffritti.pierfrancesco/handy-hands-detection-with-opencv-ac6e9fb3cec1. [Último acceso: 15 Diciembre 2019].
9. OpenCV-Python Tutorials, «Face Detection using Haar Cascades,» [En línea]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html#face-detection. [Último acceso: 15 Diciembre 2019].
10. K. Preprocessing, «Keras,» [En línea]. Available: https://keras.io/preprocessing/image/. [Último acceso: 9 noviembre 2019].
11. K. Simonyan, A. Zisserman, «arXiv.org,» [En línea]. Available: https://arxiv.org/abs/1409.1556. [Último acceso: 17 marzo 2020].
12. K. He, X. Zhang, S. Ren, J. Sun, «arXiv.org,» [En línea]. Available: https://arxiv.org/abs/1512.03385. [Último acceso: 17 marzo 2020].
13. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wonja, «arXiv.org,» [En línea]. Available: https://arxiv.org/abs/1512.00567. [Último acceso: 17 marzo 2020].
14. F. Quiroga, R. Antonio, F. Ronchetti, L. Lanzarini y A. Rosete, «A Study of Convolutional Architectures for Handshape Recognition applied to Sign Language,» de XXIII Congreso Argentino de Ciencias de la Computación, La Plata, 2017, p. 1346.
15. K. Applications, «keras.io,» [En línea]. Available: https://keras.io/applications/. [Último acceso: 17 Marzo 2019].
16. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto y H. Adam, «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,» *Arxiv,* p. 9, 2017.
17. Google, «developers.google.com,» 27 Marzo 2018. [En línea]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy. [Último acceso: 5 Septiembre 2019].
18. M. L. Mastery, «Machine Learning Mastery,» [En línea]. Available: https://machinelearningmastery.com/k-fold-cross-validation/. [Último acceso: 9 Diciembre 2019].
19. o. t. closing, «opencv tutroals closing,» [En línea]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#closing. [Último acceso: 20 Diciembre 2019].
20. K. Models, «Keras Models,» [En línea]. Available: https://keras.io/models/sequential/#evaluate_generator. [Último acceso: 15 Diciembre 2019].

Apéndice 1.

Colector del conjunto de datos (“Dataset Collector”)

Esta herramienta proporciona la interfaz gráfica que se visualiza en la Figura 13.

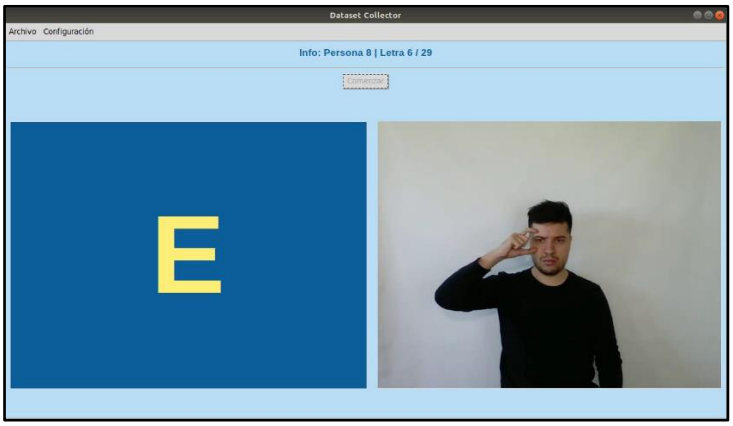

Fig. 13. Interfaz gráfica “Dataset Collector”.

Descripción de la interfaz gráfica:

- En la parte izquierda de la pantalla se muestran secuencialmente cada una de las letras (previamente configuradas) pertenecientes al alfabeto dactilológico argentino.
- En la parte derecha de la pantalla se visualizan las imágenes capturadas por la cámara web de una computadora, idealmente es en este sector en dónde se visualiza el intérprete de la seña.

Funcionamiento de la herramienta:

- El intérprete realiza la seña indicada en la parte izquierda de la pantalla. Cada 6 segundos (tiempo por defecto con condiciones de luz óptimas), las letras irán avanzando secuencialmente hasta recorrer por completo el abecedario establecido.
- Mientras el sujeto realiza las señas, el software irá generando un video por cada letra que el intérprete haya realizado.
- La herramienta creará una carpeta por cada intérprete, dentro de la cual se almacenarán los videos correspondientes a cada letra realizada.
- Cada video generado tendrá como nombre la letra que realizó el intérprete en ese momento. Esto facilitará el proceso de etiquetado de las imágenes.

Configuración de la herramienta:
Se definen los siguientes apartados para personalizar:

- **Letras**: es posible determinar aquellas letras con las que se desea trabajar. En caso de no especificarlo, por defecto se establece el abecedario completo.

- **Cantidad de personas:** permite definir la cantidad de personas que ya realizaron el proceso, de modo tal que la herramienta comience a crear carpetas a partir del número establecido. Por ejemplo, si se indicara que 5 personas realizaron el proceso, la próxima vez que seleccionemos "Comenzar", empezará a partir de la persona 6 (generando el directorio correspondiente). Por defecto el número de la persona con la que comenzará será igual a la cantidad de carpetas que haya en el directorio de videos más 1.
- **Duración letras:** tiempo que durará el video correspondiente a cada letra. Por defecto es de 6 segundos.
- **Cantidad frames por segundo:** cantidad de cuadros por segundo que contendrán los videos generados. Por defecto: 30.
- **Usar líneas guía:** genera líneas de guía sobre las imágenes tomadas por la cámara web, permitiendo centrar a las personas que utilicen la herramienta.
- **Voltear imagen (eje y):** voltea sobre el eje y la imagen que se visualiza tanto en la parte derecha de la pantalla, como en el video resultante.
- **Directorio videos:** es factible seleccionar el directorio en el que se quieren almacenar los videos generados. En caso de no seleccionar este parámetro, el software establece un directorio por defecto.

Apéndice 2.

Videos a imágenes (Videos to images)

Videos to images transforma los videos generados por la herramienta Dataset Collector en cada una de las imágenes que los componen.

Además, la herramienta coloca el nombre a cada imagen a partir de la letra indicada en el nombre del video de origen. Realizando de este modo, el proceso de etiquetado automático del "dataset". También permite descartar un número fijo preestablecido de fotogramas al inicio de cada video, esto resultó útil en la etapa de limpieza de datos.

Sumado a esto, cada persona puede indicar el número del fotograma en que inicia cada letra, permitiendo descartar una porción de los fotogramas anteriores a este umbral y el resto atribuírselos a la letra anterior (a excepción de la letra A). Por ejemplo, si se establece que la letra B comienza en el fotograma 45 para la persona 1, por defecto la herramienta asigna los fotogramas del 1 al 30 a la letra anterior y descarta los 15 restantes.

Como resultado se creará una carpeta por cada uno de los intérpretes con las imágenes de todas las señas que realizó.

Apéndice 3.

Segmentación de imágenes por color.

Funcionamiento de la herramienta desarrollada:

Cada imagen de cada directorio es transformada al espacio de color HSV, posteriormente se le aplica los filtros de color definidos y el método de closing [19]. Luego obtiene las coordenadas que encierran a la mano, para posteriormente recortar la imagen y almacenarla en un nuevo directorio.

Al finalizar el proceso se cuenta con un nuevo directorio con la misma estructura que el que recibió como entrada, pero con las imágenes segmentadas por el color definido.

Apéndice 4.

Preparar dataset (*Prepare Dataset*)

Toma las imágenes generadas por la herramienta Videos to Images y las reorganiza en un conjunto de directorios como el que se puede visualizar en la Figura 14.

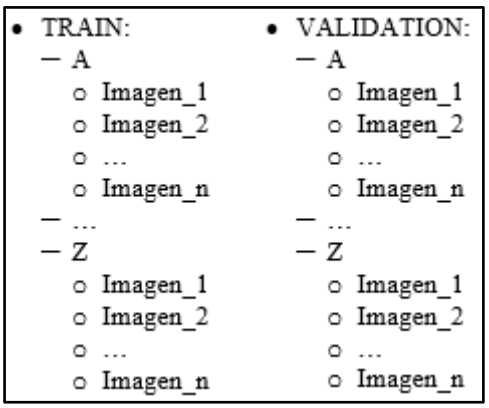

Fig. 14. Jerarquía de directorios generada por *Prepare Dataset*.

La herramienta permite seleccionar el porcentaje de la cantidad de individuos que formarán parte del conjunto de *train* y de la cantidad que formarán parte del de *validation*. Por defecto el 80% de los individuos constituirán el conjunto de *train* y el 20% restantes formarán parte del conjunto de *validation*. Esto significa que si, por ejemplo, se generaron señas hechas por 10 individuos, las realizadas por 8 de los 10 se utilizarán para entrenar el modelo y las hechas por los 2 restantes se utilizarán para validar el funcionamiento del mismo.

Apéndice 5.

Resultados experimentos realizados.

Dataset imágenes sin guantes.

Resultados obtenidos:

Red Neuronal Convolucional: Personalizada 1:

Table 5. Resultado Red Neuronal Convolucional: Personalizada 1 – Imágenes sin guantes.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
20	34.597	8.354	0,9896	0,5568

En este experimento se emplearon 12 personas para entrenar el modelo y 2 para validarlo. Se utilizó *Data Augmentation*. Los resultados obtenidos evidencian sobre-entrenamiento.

Red Neuronal Convolucional: Personalizada 2:

Table 6. Resultado Red Neuronal Convolucional: Personalizada 2 – Imágenes sin guantes.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
15	34.648	6.440	0,9758	0,3567

En este experimento se emplearon 11 personas para entrenar el modelo y 2 para validarlo. Se utilizó *Data Augmentation*. Los resultados obtenidos evidencian sobre-entrenamiento.

***MobileNet* – Validación Cruzada:**

Resultados entrenamiento – Validación Cruzada:

Table 7. Resultado Validación Cruzada Entrenamiento *MobileNet* – Imágenes sin guantes.

Dataset	Cant. Datos Entrenamiento	Cant. Datos Validación	Épocas Entrenamiento	*Accuracy* Entrenamiento	*Accuracy* Validación
1	45.580	6.509	57	0,9978	0,9075
2	45.719	6.370	27	0,9928	0,7408
3	43.668	8.421	39	0,9970	0,8115
Promedio				0,9959	0,8199

Resultados Test:

Table 8. Resultado Test *MobileNet* – Imágenes sin guantes.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
70	52.089	3.844	0,9995	0,6485

Se evidencia sobre-entrenamiento en los resultados arrojados luego de evaluar el modelo en el conjunto de test.

Dataset imágenes sin guantes recortadas.

Resultados obtenidos:

Red neuronal Convolucional: Personalizada 3:

Table 9. Resultado Red Neuronal Convolucional: Personalizada 3 – Imágenes sin guantes recortadas.

Cant. Épocas	Data Augmentation	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
50	NO	34.648	6.440	0,7100	0,5000
50	SI	34.648	6.440	0,6929	0,4778

Los resultados obtenidos evidencian que en este modelo el uso de técnicas de *Data Augmentation*, empeoran los resultados.

***VGG-16*: Transfer Learning:**

Table 10. Resultado *VGG-16: Transfer Learning* – Imágenes sin guantes recortadas.

Cant. Épocas	Data Augmentation	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
9 (paciencia = 5)	NO	34.648	6.440	0,9214	0,5878
50	SI	34.648	6.440	0,5763	0,4837

Es interesante observar como conforme el entrenamiento avanza, en el experimento con *Data Augmentation*, los valores de la métrica disminuyen, esto puede observarse en la Figura 15, para reducir este fenómeno, en el experimento sin *Data Augmentation* se empleó el modelo utilizando regularizaciones y *BatchNormalization*.

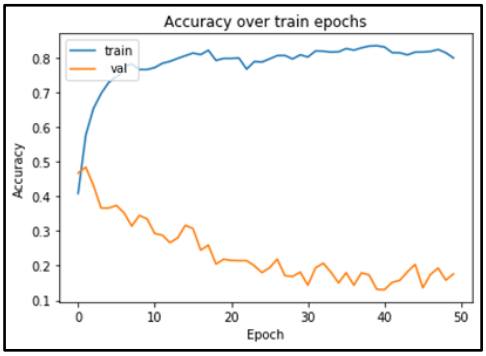

Fig. 15. Gráfico Resultado *VGG-16: Transfer Learning* con *Data Augmentation*.

***ResNet-50*:**

Table 11. Resultado *ResNet-50* – Imágenes sin guantes recortadas.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
34 (paciencia=15)	34.648	6.440	0,9968	0,8798

Los resultados evidencian un comportamiento aceptable de este modelo sobre el conjunto de datos de imágenes sin guantes recortadas.

***MobileNet*:**

Table 12. Resultado *MobileNet* – Imágenes sin guantes recortadas.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
40 (paciencia=15)	34.648	6.440	0,9966	0,8901

Como se puede observar, este modelo es el que mejores resultados presenta hasta el momento sobre el conjunto de imágenes sin guantes recortadas.

***Inception V3*:**

Table 13. Resultado *Inception V3* – Imágenes sin guantes recortadas.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
31 (paciencia = 15)	34.648	6.440	0,9951	0,8829

Los resultados evidencian un buen comportamiento de este modelo sobre el conjunto de imágenes sin guantes recortadas.

Como se puede observar, los resultados presentados por los modelos *ResNet-50*, *MobileNet* e *Inception V3* son aceptables, frente a este escenario se realiza una comparativa de tiempo de predicción entre estos tres modelos a fin de encontrar aquel que se desempeñe mejor en este aspecto. Los resultados de dicha comparativa pueden encontrarse en el **Apéndice 6**, y a partir de los valores allí obtenidos se opta por la utilización de uno de los modelos planteados en el resto de "*Datasets*".

***MobileNet* – Validación Cruzada:**

Resultados entrenamiento – Validación Cruzada:

Table 14. Resultados Validación Cruzada Entrenamiento *MobileNet* – Imágenes sin guantes recortadas.

Dataset	Cant. Datos Entrenamiento	Cant. Datos Validación	Épocas Entrenamiento	*Accuracy* Entrenamiento	*Accuracy* Validación
1	45.633	6.509	39	0,9978	0,9246
2	45.772	6.370	54	0,9996	0,9019
3	43.721	8.421	23	0,9970	0,8228
Promedio				0,9981	0,8831

Resultados Test:

Table 15. Resultados Test MobileNet – Imágenes sin guantes recortadas.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
84 (paciencia = 25)	52.142	3.844	0,9999	0,8205

Dataset imágenes segmentadas por color.

Resultados obtenidos:

Red Neuronal Convolucional: Personalizada 4:

Resultados de las pruebas sobre un subconjunto de datos:

Table 16. Resultados Red Neuronal Convolucional: Personalizada 4 – Imágenes segmentadas por color.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Validación	*Accuracy* Entrenamiento	*Accuracy* Validación
15	34.803	8.628	0,9968	0,9080

Resultados entrenamiento – Validación Cruzada:

Table 17. Resultados Validación Cruzada Entrenamiento Red Neuronal Convolucional: Personalizada 4 – Imágenes segmentadas por color.

Dataset	Cant. Datos Entrenamiento	Cant. Datos Validación	Épocas Entrenamiento	*Accuracy* Entrenamiento	*Accuracy* Validación
1	46.131	6.472	15	0,9971	0,9620
2	46.015	6.588	15	0,9917	0,9344
3	44.240	8.363	21	0,9954	0,9693
Promedio				0,9947	0,9652

Resultados Test:

Table 18. Resultados Test Red Neuronal Convolucional: Personalizada 4 – Imágenes segmentadas por color.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
46 (paciencia = 10)	52.603	3.796	0,9986	0,9584

Dataset imágenes sin guantes recortadas 2.

Resultados obtenidos:

***MobileNet* – Validación Cruzada:**

Resultados entrenamiento – Validación Cruzada:

Table 19. Resultados Validación Cruzada Entrenamiento *MobileNet* – Imágenes sin guantes recortadas 2.

Dataset	Cant. Datos Entrenamiento	Cant. Datos Validación	Épocas Entrenamiento (paciencia = 15)	*Accuracy* Entrenamiento	*Accuracy* Validación
1	45.012	6.509	33	0,9979	0,9306
2	45.151	6.370	32	0,9982	0,9144
3	43.133	8.388	28	0,9969	0,8991
Promedio				0,9977	0,9147

Resultados Test:

Table 20. Resultados Test *MobileNet* – Imágenes sin guantes recortadas 2.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
69 (paciencia = 15)	51.521	3.844	0,9998	0,7586

Dataset imágenes segmentadas por la piel.

Resultados obtenidos:

***MobileNet* – Validación Cruzada:**

Resultados entrenamiento – Validación Cruzada:

Table 21. Resultados Validación Cruzada Entrenamiento para el experimento *MobileNet* – Imágenes segmentadas por la piel.

Dataset	Cant. Datos Entrenamiento	Cant. Datos Validación	Épocas Entrenamiento	*Accuracy* Entrenamiento	*Accuracy* Validación
1	44.858	6.509	27 (paciencia = 15)	0,9970	0,9192
2	44.997	6.370	14 (paciencia = 5)	0,9952	0,8951
3	42.977	8.390	15 (paciencia = 5)	0,9963	0,8552
Promedio				0,9962	0,8898

Resultados Test:

Table 22. Resultados Test *MobileNet* – Imágenes segmentadas por la piel.

Cant. Épocas	Cant. Datos Entrenamiento	Cant. Datos Test	*Accuracy* Entrenamiento	*Accuracy* Test
20 (paciencia = 10)	51.367	3.844	0,9990	0,8096

Apéndice 6.

Comparativa de tiempo: *ResNet-50, MobileNet* e *Inception V3.*

A continuación, se presentará el tiempo que le insume a cada modelo ser evaluado sobre el mismo conjunto de test utilizando el método *evaluate_generator* [20].

Se presenta esta comparativa para fundamentar la elección del modelo, ya que los resultados obtenidos en la métrica *Accuracy* sobre el conjunto de validación fue similar para los 3 modelos.

Las especificaciones técnicas del equipo donde se realizó la prueba son las siguientes:

- Procesador: Intel Core i5 6200u
- Memoria RAM: 16 GB 2133 Mhz
- GPU: NVIDIA GeForce 940MX 2GB

Durante la ejecución de cada algoritmo se monitorearon las condiciones de procesamiento del equipo a fin de garantizar igualdad de condiciones entre los mismos.

El tamaño del conjunto sobre el que se evaluó el modelo es de 3.844 imágenes pertenecientes al "*Dataset*" imágenes sin guantes recortadas.

Resultados obtenidos:

Table 23. Comparativa de tiempo: *ResNet-50, MobileNet e Inception V3*.

Modelo	Tiempo Total (seg.)	Tiempo por imagen (seg.)	FPS resultantes
ResNet-50	167	0.043	23
MobileNet	55	0.014	69
Inception V3	168	0.044	22

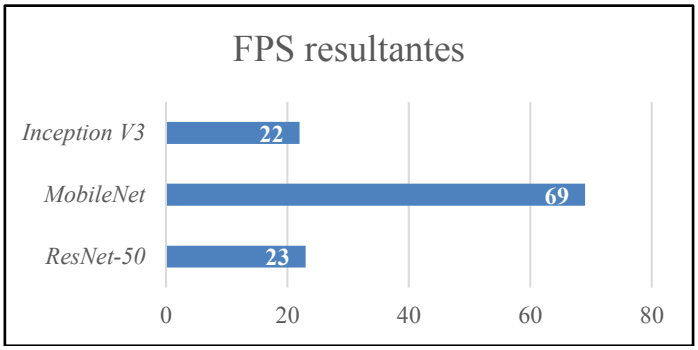

Fig. 16. Comparativa de tiempo: *ResNet-50, MobileNet e Inception V3*.

El valor de los FPS resultantes sería el expresado en la tabla en caso de que el tiempo por imagen correspondiese únicamente con la predicción, sin embargo, en este caso además de predecir, se calcula el valor de la métrica. No obstante, se incluyó en la tabla ya que ofrece una buena medida de referencia.

Conclusión:

A partir de los resultados obtenidos, se decide emplear el modelo *MobileNet* en los experimentos sobre los "*Datasets*" debido a que es el que se desempeña con mayor rapidez, permitiendo su aplicación en escenarios de "tiempo real", como los que se requieren en el presente trabajo. El único conjunto de imágenes que no emplea este modelo es el "*Dataset*" de imágenes segmentadas por color, ya que las imágenes que lo componen poseen un mayor pre-procesado con respecto al resto, lo que permite emplear modelos con menor cantidad de parámetros, como la Red Neuronal Convolucional: Personalizada 4.